### **Task 5**

**Machine Learning**

Upload .py or Ipynb extension file on GitHub public repo “100DaysofBytewise" and share the link in the submission form by 24 June 2024.

1. **Implement a linear regression model to predict housing prices based on a given dataset.**

**Expected Output:**

* 1. **Load a dataset the Boston Housing dataset.**
  2. **Train a linear regression model.**
  3. **Print the model's coefficients and intercept.**
  4. **Predict housing prices on a test set and print the mean squared error.**
  5. **Visualize the regression line and data points.**

**Code:**

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import fetch\_california\_housing

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

california = fetch\_california\_housing()

X = california.data

y = california.target

feature\_names = california.feature\_names

import pandas as pd

df = pd.DataFrame(data=X, columns=feature\_names)

df['Target'] = y

print("Data (first 5 rows):")

print(df.head())

sns.pairplot(df, vars=feature\_names, diag\_kind='kde')

plt.suptitle('Pairplot of California Housing Dataset')

plt.show()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression()

model.fit(X\_train, y\_train)

print("\nCoefficients:", model.coef\_)

print("Intercept:", model.intercept\_)

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

print("Mean Squared Error:", mse)

sns.histplot(y\_pred, bins=30, kde=True)

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, color='blue', label='Actual vs Predicted')

plt.plot([y.min(), y.max()], [y.min(), y.max()], '--', color='red', linewidth=2, label='Ideal Predictions')

plt.title('Actual vs Predicted Housing Prices')

plt.xlabel('Actual Prices')

plt.ylabel('Predicted Prices')

plt.legend(loc='upper left')

plt.grid(True)

plt.tight\_layout()

plt.show()

1. **Build a decision tree classifier to classify iris flower species.**

**Expected Output:**

* 1. **Load the Iris dataset.**
  2. **Train a decision tree classifier.**
  3. **Print the classification report and confusion matrix.**
  4. **Visualize the decision tree.**

Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import classification\_report, confusion\_matrix

# Load the Iris dataset

iris = load\_iris()

A = iris.data

B = iris.target

feature\_names = iris.feature\_names

target\_names = iris.target\_names

# Create a DataFrame for easier plotting

iris\_df = pd.DataFrame(data=A, columns=feature\_names)

iris\_df['target'] = B

# Split the data into training and test sets

A\_train, A\_test, B\_train, B\_test = train\_test\_split(A, B, test\_size=0.21, random\_state=42)

# Initialize the decision tree classifier

dt\_classifier = DecisionTreeClassifier(random\_state=42)

# Train the classifier

dt\_classifier.fit(A\_train, B\_train)

# Predictions on the test set

y\_pred = dt\_classifier.predict(A\_test)

# Print confusion matrix and classification report

print("Confusion Matrix:")

print(confusion\_matrix(B\_test, y\_pred))

print("\nClassification Report:")

print(classification\_report(B\_test, y\_pred, target\_names=target\_names))

# Pairplot of the dataset

sns.pairplot(iris\_df, vars=feature\_names, diag\_kind='kde')

plt.suptitle('Pairplot of Iris Dataset', y=1.02)

plt.show()

# Visualize the decision tree

plt.figure(figsize=(20, 15))

plot\_tree(dt\_classifier, filled=True, feature\_names=feature\_names, class\_names=target\_names)

plt.title("Decision Tree Classifier - Iris Dataset")

plt.show()